Qiong Hu (405065032)
Zihao Zou (005349580)
Gaofang Sun (104853165)

Project 4 Report

Graph Algorithm

ECE 232E
Large Scale Networks
June 12, 2020

1. Stock Market

1. Return correlation

QUESTION 1:

The bounds of ρ_{ij} is: $-1 \leqslant \rho_{ij} \leqslant 1$. The lower bound -1 is reached when $r_j(t)$ is proportional to $r_i(t)$ but has the opposite sign for every t. The upper bound 1 is reached when $r_j(t)$ is proportional to $r_i(t)$ for every t. The proof is as follows:

When $r_j(t) = kr_i(t)$, where k is a constant over t, then

$$\begin{aligned}\rho_{ij} &= \frac{\langle r_i(t)r_j(t)\rangle - \langle r_i(t)\rangle \langle r_j(t)\rangle}{\sqrt{\left(\langle r_i(t)^2\rangle - \langle r_i(t)\rangle^2\right)\left(\langle r_j(t)^2\rangle - \langle r_j(t)\rangle^2\right)}} \\ &= \frac{k\langle r_i(t)^2\rangle - k\langle r_i(t)\rangle^2}{\sqrt{k^2\left(\langle r_i(t)^2\rangle - \langle r_i(t)\rangle^2\right)^2}} \\ &= \frac{k}{|k|}\cdot\frac{\langle r_i(t)^2\rangle - \langle r_i(t)\rangle^2}{|\langle r_i(t)^2\rangle - \langle r_i(t)\rangle^2|} \\ &= \operatorname{sign}(k).\end{aligned} \tag{1}$$

The reasons why using log-normalized return ($r_i(t)$) instead of regular return ($q_i(t)$) to calculate the correlation ρ_{ij} are as follows:

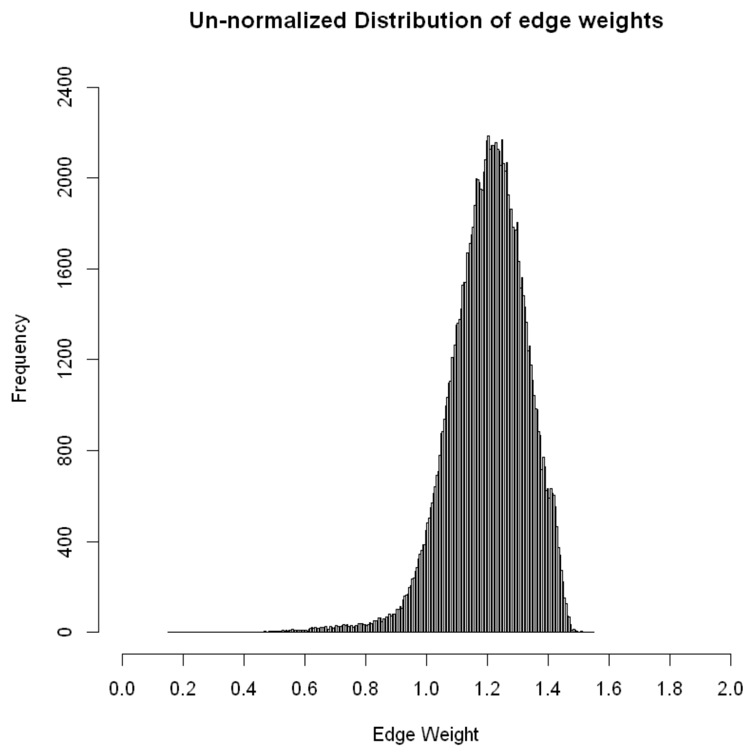
- Empirically, the change of $p_i(t)$ over a period of $[t-1, t]$ is trivial compared to $p_i(t-1)$, so typically $q_i(t)$ would be a small value around 0 within range of $[-1, 1]$. Log normalization maps this small value to a range of $(-\infty, \log 2]$, magnifying the negative results. It means log-normalized return $r_i(t)$ ensures that, when the closing stock-return at the t^{th} day is lower than $(t-1)^{th}$ day, the return value would be more significant compared to when it's higher.
- Log-normalization method also enables the addition of return value over a period of time, which could otherwise be tricky for regular return $q_i(t)$.

$$\sum_{t=t_0}^{t} r_i(t) = \sum_{t=t_0}^{t}\log(1+q_i(t)) = \sum_{t=t_0}^{t}\log\left(\frac{p_i(t)}{p_i(t-1)}\right) = \log p_i(t) - \log p_i(t_0-1). \tag{2}$$

2. Constructing correlation graphs

QUESTION 2:

In this part, we calculated the edge weights based on the given expressions, and show the un-normalized distribution in the following diagram.

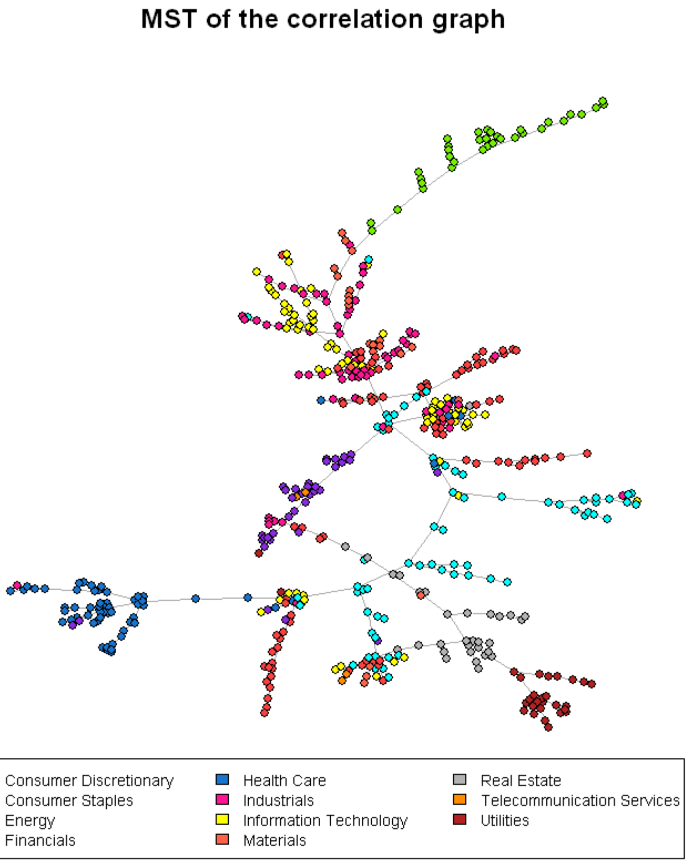


From the diagram, we can see a peak at around weight $w = 1.2$, and a large amount of edge weights fall in between about 1.0 to 1.4. We also notice an asymmetric trend where there is a longer tail in the left side of the peak than to the right side of the peak. Since $w_{ij} \to 0$ means $\rho_{ij} \to 1$, and $w_{ij} \to 2$ means $\rho_{ij} \to -1$, the histogram suggests that the stocks are more likely to have a similar trend between each other than opposite trend.

3. Minimum spanning tree (MST)

QUESTION 3:

We extract the MST of the correlation graph and plot the result as below. We observed that the nodes on the same branch tend to have a same color, indicating that these stock nodes belong to a same sector. The reason of this Vine cluster feature is that, stocks from the same sector often have a similar fluctuation trend when facing external influences, so they tend to have a higher correlation value among them, leading to a low edge weight, therefore according to MST algorithm, they would be easier to be clustered together.



4. Sector clustering in MST's

QUESTION 4: We calculated α for the MST graph from Question 3 in two cases:

1. When $P(v_i \in S_i) = \dfrac{|Q_i|}{|N_i|}$, $\alpha = 0.8140095$.

2. When $P(v_i \in S_i) = \dfrac{|S_i|}{|V|}$, $\alpha = 0.1145652$.

In the first case, a good sector clustering method should return an α that is close to 1, because for each node, $\frac{|Q_i|}{|N_i|}$ calculates the possibility of node i's neighbors to be in the same sector as i. In MST, since each node has only one neighbor, this value would be either 0 or 1. By calculating the average of all nodes, MST graph shows a good performance in this sector clustering task with an average accuracy of about 80%.

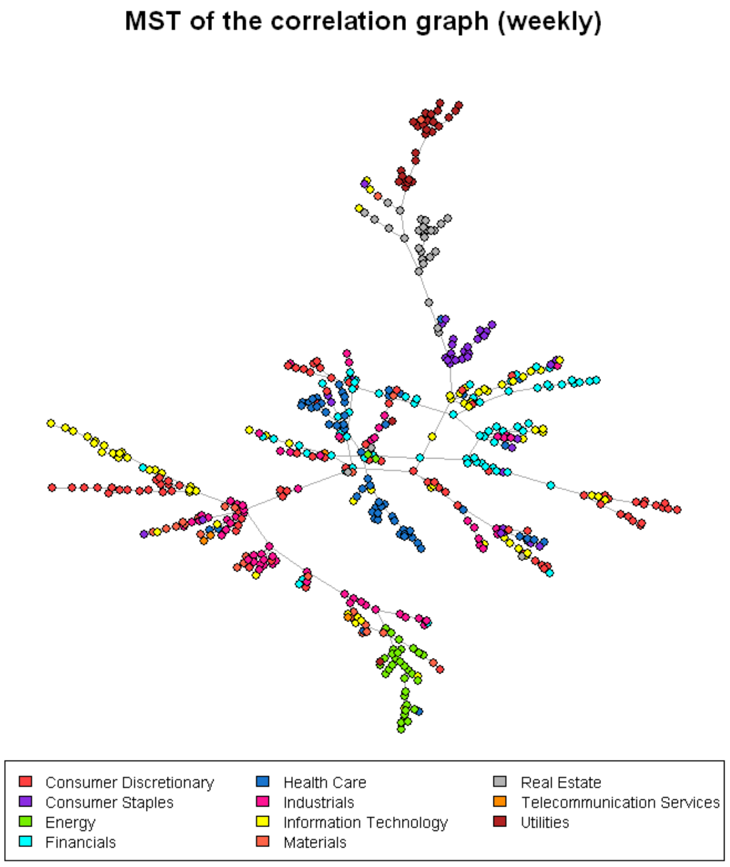
In the second case, α value appears to be a lot smaller than the first case, because the term $\frac{|S_i|}{|V|}$ does not actually reflect the performance of each individual nodes, instead, it is more like an evaluation for sectors. α in the second case can be rephrased as:

$$\alpha = \frac{1}{|V|^2} \sum_{S_i} |S_i|^2, \tag{3}$$

where $|S_i|$ is the number of nodes in the sector i. So the first definition of α would be a more sufficient evaluation method for sector clustering.

5. Correlation graphs for weekly data

QUESTION 5: We extracted the MST from the correlation graph based on weekly data, and the resulting figure is plotted below.



Compared this figure with the figure from Question 3, we observed a similar pattern where MST branches tend to have more nodes with a same color, but the pattern is less conspicuous with a lot more noises. A possible reason is that by reducing the raw data to about 20%, the time-dependent stock vibrations have larger fluctuations and are more easily influenced by sector-independent factors, thus stock companies in the same sector show a less similar trend, resulting in smaller correlation values larger edge weights. Therefore, the new MST figure has a worse performance in sector clustering.

Also, using the evaluation metric from Question 4, the two α's for the new MST are 0.7201194 and 0.1145652, respectively. In the second case, α value remains the same because it is only relevant to the sector of each node themselves, without reflecting the graph structure, therefore, it's insufficient for evaluation, as we have discussed in Question 4. In the first case, the average possibility of a node to be in the same sector as its only neighbor reduces, which supports our conclusion that the new MST with weekly data shows a worse performance.

2. Let's Help Santa!

In this part, we use network theory to facilitate the delivery problem with a vast amount of statistics about transportation dynamics. Especially, we first built the graph based on the Uber data in San Francisco area, and selected out the GCC, and found an approximation solution for the travelling salesman problem(TSP) on the graph. In the end, we calculated the traffic flow between two given locations.

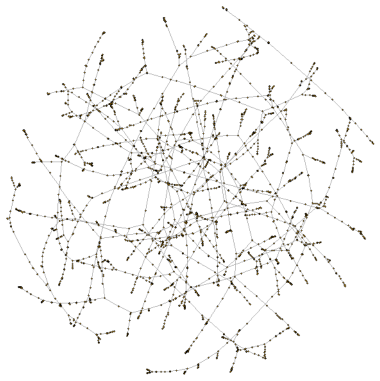
2. Build Your Graph

We went to "Uber Movement" website and download data of Travel Times by Month (All Days), 2019 Quarter 4, for Los Angeles area1. We build a graph in which nodes correspond to locations, and undirected weighted edges correspond to the mean traveling times between each pair of locations. The graph will contain some isolated nodes (extra nodes existing in the Geo Boundaries JSON file) and a few small connected components. We removed such nodes and just keep the largest connected component of the graph. And duplicates edges were merged and the final weight is the average of duplicated weights.

QUESTION 6: The remaining giant connected component is called G The number of nodes in G is 2649. The number of edges in G(gcc) is 1003858.

3. Traveling Salesman Problem

QUESTION 7: In this question, we construct a minimum spanning tree of graph G.



Endpoint locations of edge 2 are
"-118.555078635802, 34.2813026790123" and "-118.562330458065, 34.2855382451613".
Distance in km is: 0.932194803979754
Endpoint locations of edge 200 are
"-118.463208647541, 34.1514969590164" and "-118.454772666667, 34.1562754666667".
Distance in km is: 1.07618440556906
Endpoint locations of edge 1500 are
"-118.096356434783, 33.9760686956522" and "-118.090718325581, 33.9720804883721".
Distance in km is: 0.766575938451483
From the table above, we can find that the results are intuitive because the locations are very close to each other. This result is consistent with the minimum spanning tree theory.

QUESTION 8: After the random samplings of 1000 triangles on the graph, the percentage of triangles is 96.7%.

QUESTION 9: In this problem, we want to find an approximated solution for the traveling salesman problem (TSP) on our graph G, and compare the difference between the approximated and optimal TSP cost. Since TSP cost problem is based on the high percentage of triangles that satisfy the triangle inequality, we aim to use the approximation algorithm and to get approximated TSP cost as close to the optimal TSP cost as possible. To calculate the optimal TSP cost is NP hard, but we can use MST cost instead because we have the relationship below:

$$\rho = \frac{\text{Approximate TSP cost}}{\text{Optimal TSP cost}} < \frac{\text{Approximate TSP cost}}{\text{MST cost}}.$$

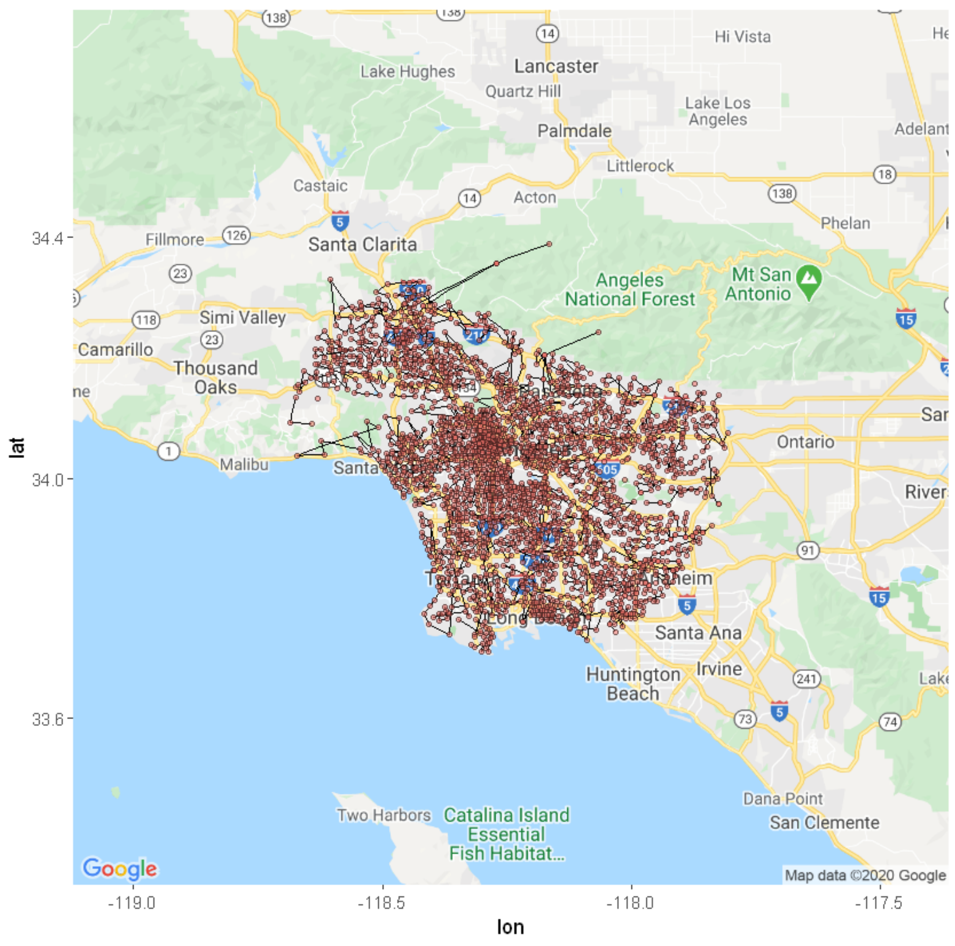
The approximate TSP tour sequence varies as we change the starting point of the travel, and the costs vary accordingly. Thus, we randomly selected 50 different starting nodes, we pick the route with the maximum cost. Based on our results, MST cost is 289315.675, the maximum approximate TSP we got is 473671.17, and the upper bound is 1.76030611494243. Empirically, however, the relationship between the costs are listed below:

$$\text{MST cost} \leqslant \text{optimal TSP cost} \leqslant \text{approximate TSP cost} \leqslant 2 \times \text{MST cost}.$$

Thus, the theoretical upper bound of ρ is 2 as it changes between 1 and 2.

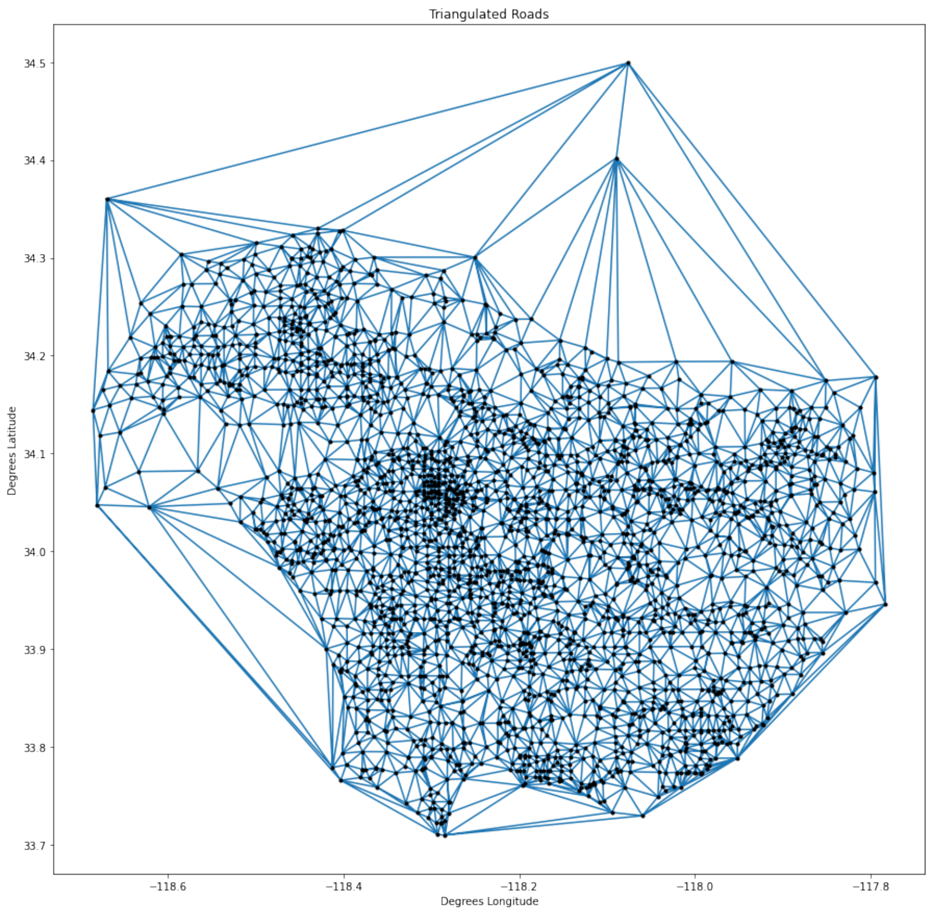
QUESTION 10:

Based on the calculated permutation of the nodes, we can plot the trajectory that Santa can travel with the optimized travelling time. We exclude the route between nodes do not exist in the original graph, and count the shortest path between the sequential node pair as the real path. (We also plotted the trajectory without excluding "imaginary" routes in the attached notebook, and we clearly see the difference in routes between the same group of nodes.) The result looks intuitively right, as bigger amount of nodes gather around the large cities, and all the paths are relatively short without long paths between nodes that are far away from each other.



5. Estimate the Roads

QUESTION 11: In this part, the Delaunay triangulation algorithm is used to generate a triangular mesh graph.The Delaunay triangulation algorithm is an algorithm that uses all the points in the mesh to create triangles such that no other vertex lies inside a circumscribed circle of each triangle. The following is the plotted triangular mesh of the map.



The above graph is produced with nodes as difference location coordinates from the JSON dataset. The edges are generated with the Delaunay triangulation algorithm. In this graph, virtual edges are generated disregarding the actual geography. Hence, we can see edges across the sea or mountain with respect to the actual map.

6. Calculate Road Traffic Flows

QUESTION 12: To calculate the flow of each road, we must first calculate the velocity, v, for each road. This is done simply by dividing the Euclidean distance found with two coordinates by the average travel time provided in the dataset. Afterward, to get the traffic flow, which is car per hour, we can first determine the opposite, the time each car takes to travel through a point. To find the time each car takes to flow through a point. We can use the following equation

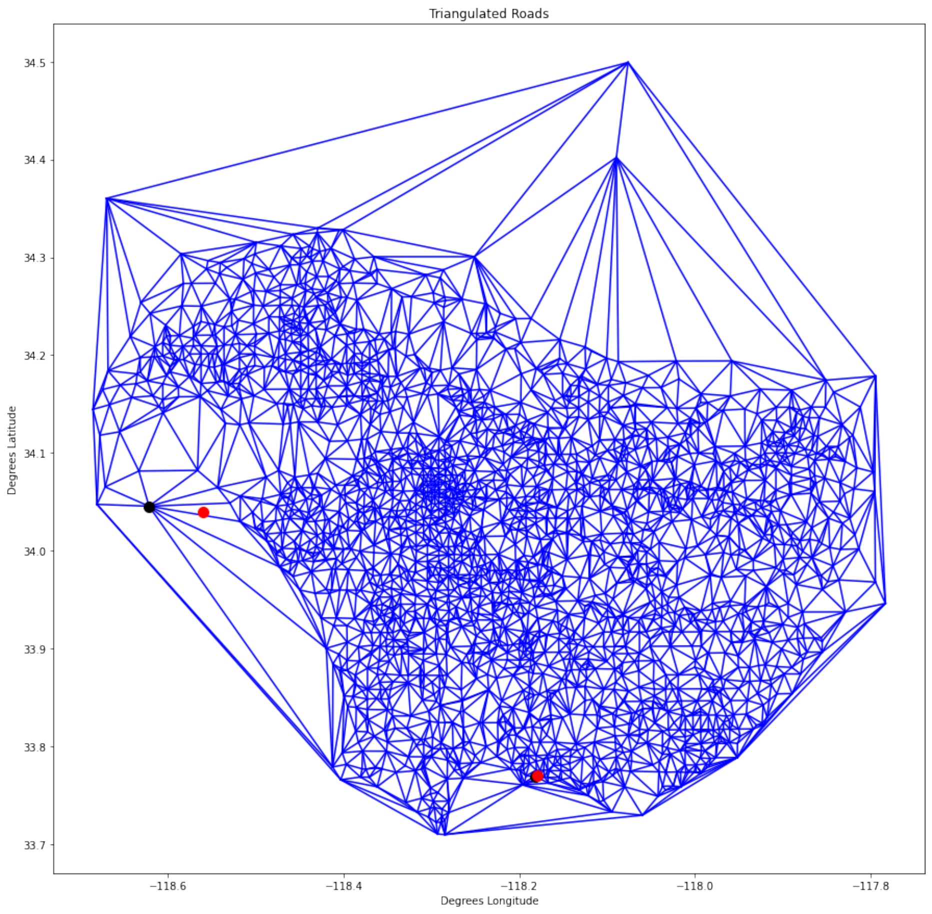
$$\Delta t = \frac{0.003}{v} + \left(\frac{2}{3600}\right) \text{ hours/car} \tag{4}$$

In the above equation, 0.003 is the length of each car. Dividing the length of each car by the velocity. After adding two seconds of safety distance, we obtain the time needed for a car to travel through a point. Now, the flow in car per hour is simply taking the inverse of the above equation and times 2 to account for two lanes.

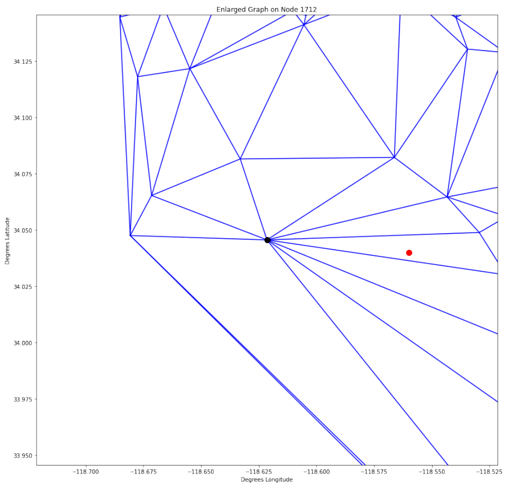
$$\text{flow} = 2 * \frac{1}{\Delta t} \text{ cars/hour} \tag{5}$$

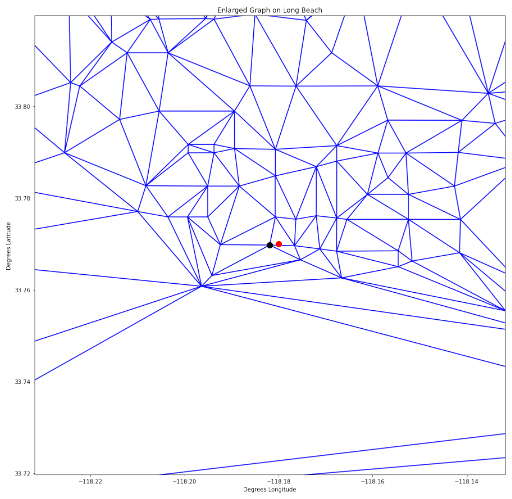
7. Calculate Max Flow

QUESTION 13: To calculate the maxflow between Malibu and Long Beach, we must first find the movement ids with respect to the given coordinates. Since each given coordinate locates inside a triangle, we can use the find simplex method in the Scipy Delaunay function to find the triangle. After looking up those triangles, we use a point closest in the triangle as the source and sink node of the max flow problem. The following is the triangular mesh plotted with the two nodes shown. The red points are the coordinates provided in the project statement.



The triangulation graph with movement ids as nodes and flow time as weights is then imported to R. Using the max flow function and edge disjoint path function, we are able to calculate the max flow and the number of disjoint paths from Malibu to Long Beach. The calculated max flow from Malibu to Long Beach is 12296.96 and the number of disjoint paths is 5. To confirm if the number of disjoint paths makes sense, we plotted the map on an enlarged scale.

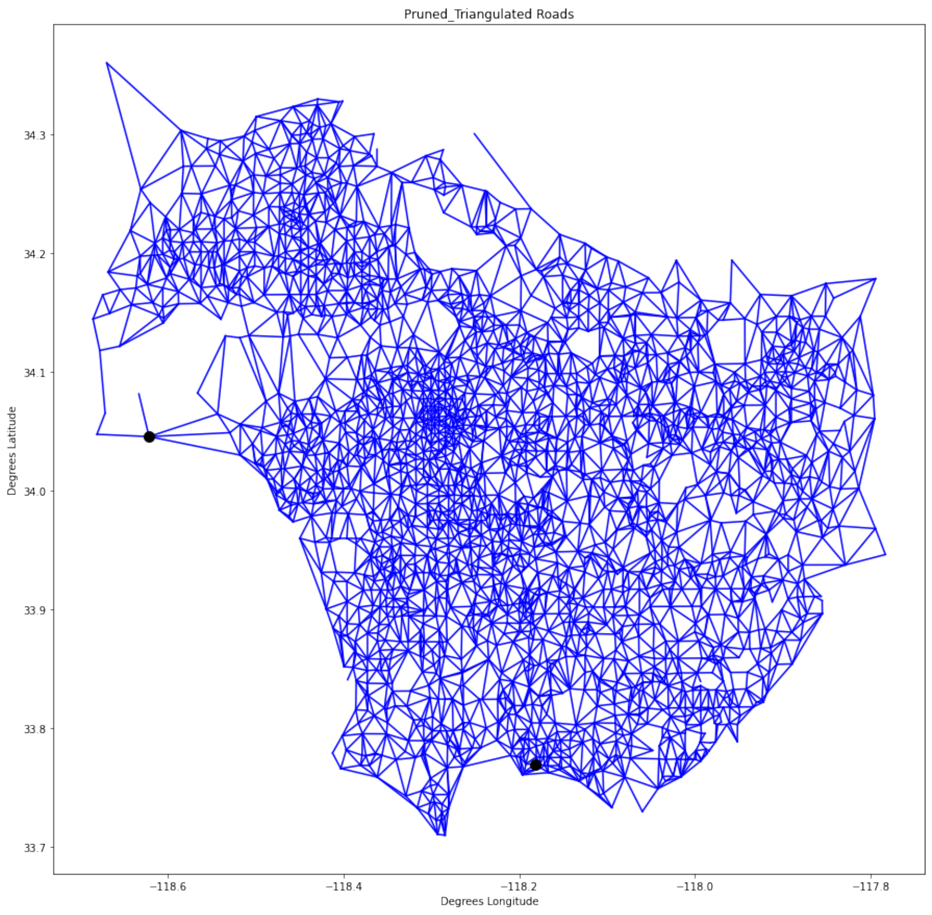




From the above figures, although many edges extend out from the node by Malibu, we can see that only 5 edges extend from the Long Beach nodes. Hence, the result of 5 disjoint paths found between the two nodes is reasonable.

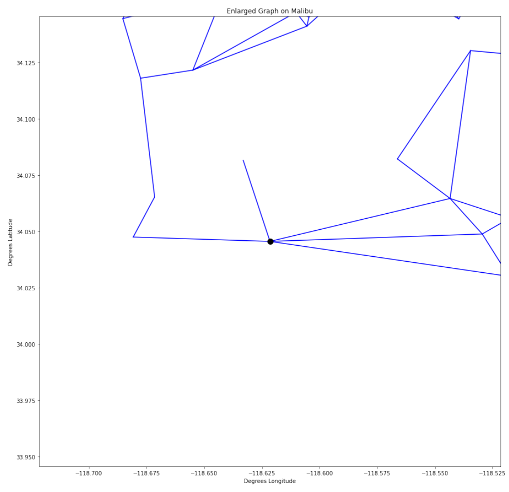
8. Prune Your Graph

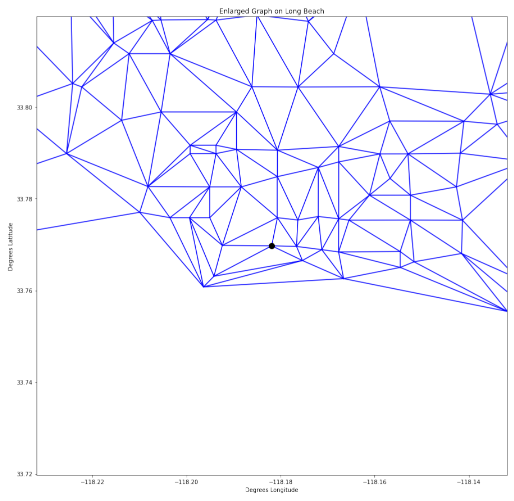
QUESTION 14: Since the triangulation algorithm plots virtual roads, unreal links such as across the ocean or a mountain can be removed to increase the fidelity. To do this, the mean traveling time threshold is introduced. Since the average travel time is set as a weight for each edge, edges with a weight larger than the threshold is removed. The threshold is set to 10 minutes. The following is the pruned graph.



After pruning, edges across mountains and ocean are removed indeed. The resulting graph shows the concavity of the coastline similar to the actual geographical map.

QUESTION 15: After removing the unreal edges, the max flow and number of disjoint paths are calculated again. After removing the edges, the max flow is 12073.25 and the number of disjoint paths is 4. Again, the enlarged map is shown below.





From the above maps, it makes sense the the number of disjoint paths has reduced to 4. Since the unreal edges are deleted, edges from Malibu that go across the ocean or the mountain are removed. This left Malibu to have only 4 edges left. Therefore, the number of 4 disjoint paths makes sense.

9. Define Your Own Task

Task Statement:

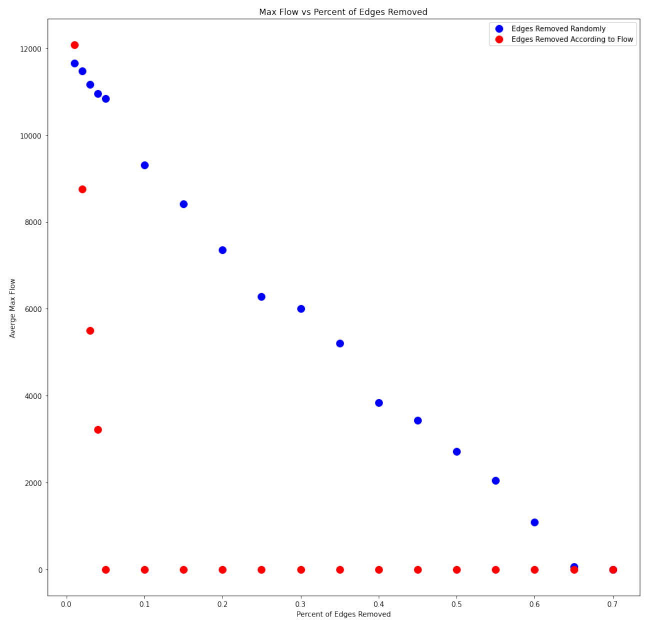
Lately, we have been facing unexpected situations such as the pandemic and the protests against racism. Both of these situations can influence traffic and movement. To be specific, the stay-at-home order limits the movement of people to reduce the spread of virus. Furthermore, in recent protests, protesters occupied the 405 freeway for a period of time to make a statement, which interferes with the normal traffic. These situations would all cast some effects on the flow between places.

In this part of the project, we aim to simulate the influence of these interference by examining if we can block the flow of traffic by deleting certain amount of randomly selected edges. Also, we want to estimate how many edges could be blocked before paralyzing the traffic in Los Angeles and which ones are more significant.

Approach and Result:

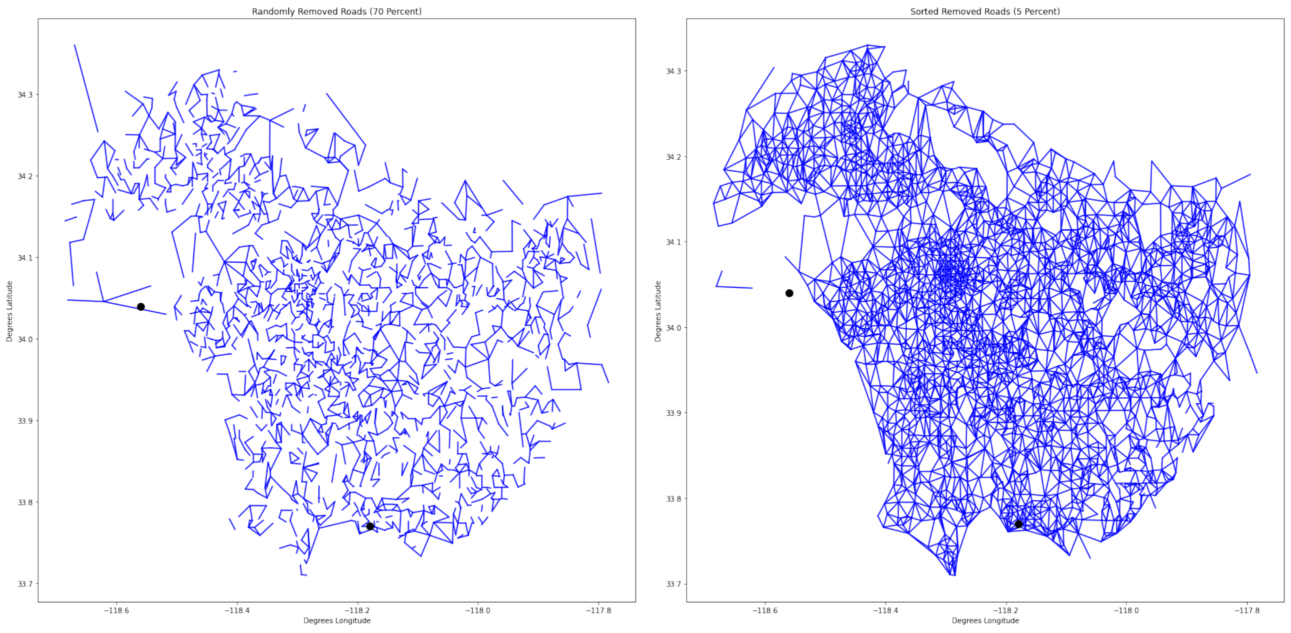
Since the task is to determine how many edges need to be cut of in order to stop the traffic flow between the source and the sink node, the minimum amount of edges needed to be remove can be modeled as a min. cut problem. As discussed in the lecture, we can treat the min. cut problem as a max flow problem. The min cut capacity to separate two groups is equal to the max flow. Therefore, if we remove enough edge to reduce the max flow to zero, we will achieve the min cut condition. Furthermore, we would want to examine what is the best way to remove edges. To simulate the traffic, two approaches are examined. The first is to randomly delete edges in the graph and calculate the flow and the number of disjoint paths between two points. Since we are randomly removing edges, 100 trials are performed at each percentage and the ensembled average is used for the result. The second approach is to delete the edges according to the flow of each edge. The traffic flow model is based on the previous part. The edges are sorted according to the weight and the removal process starts with the edge containing the largest weight. For both approaches, the percentages of edges removed are $[1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70]$.

The following is the result represented by the decrease in flow.



From the max flow to zero, the first approach, randomly removal, takes about 70 percent of the edges removed. A linear behavior between the drop of flow and the percent of edges removed is displayed. On the other hand, the flow dropped significantly for every percent of the edges removed in the second approach. Within the top 5 percent of the edges with largest weights removed, the flow has decreased to zero.

The following are the maps plotted when the max flow is decreased to zero. The left side is removing edges randomly and the right one is removing edges according to the weights (travel flow).



For the randomly removal case, the plot shows the graph when 70 percent of the edges removed.

On the other hand, the plot on the right shows the case of removal according to weights and only top 5 percent of the edges are removed. We notice that, for the case in the right, edges are removed near the Malibu node. This suggests that the edges near Malibu have most of the largest flows, which makes sense since the Pacific highway is located there.

Moreover, for future studies, we can also try to delete edges base on the community of the graph. The approach for this method will be first detecting multiple communities using different techniques with respect to measures such as modularity scores or walktrap. Then, we can reduce the graph to only have the centroid for each community and the community based average flow from one centroid to another. Lastly, we can remove centroids or communities and reduce the flow to examine the significance of each community to the traffic flow. This will be an interesting approach as well.